WHAT IS CLAIMED IS:

1. A compiler parallelizing schedule method comprising the steps of:

calculating a priority value of each of commands based upon mutual dependence between commands;

calculating a reverse priority value corresponding to the shortest command ending time for each of the commands;

weighting each of the commands based upon the reverse priority value; and

calculating a new priority value for each of the commands based upon the weighting value applied to each of the commands and the priority value of each of the commands.

2. A compiler parallelizing schedule method comprising the steps of:

calculating a priority value of each of commands based upon mutual dependence between commands;

checking to see whether or not there is any delay between the commands having the same priority value due to an issue limitation;

when any delay exists due to an issue limitation, calculating a reverse priority value corresponding to the shortest command ending time for each of the commands;

weighting each of the commands based upon the reverse priority value;

calculating a new priority value for each of the commands based upon the weighting value applied to each of the commands and the priority value of each of the commands; and

determining an issuing order of the commands based upon the new priority values, thereby slot-mapping the respective commands.

3. The compiler parallelizing schedule method according to claim 2,

wherein a group of the commands, each having any delay due to an issue limitation, is defined as an optimizing target group, a common precedent command of a plurality of commands contained in the optimizing target group is defined as a neck command, and the reverse priority value is found between the neck command and the optimizing target group.

4. The compiler parallelizing schedule method according to claim 3,

wherein the weighting values include a first weighting value that is applied to the commands from the optimizing target group to the neck command and a second weighting value that is applied to precedent commands preceding the neck command.

5. The compiler parallelizing schedule method according to claim 4,

wherein with respect to a plurality of commands contained in the optimizing target group, an order of priority is set in an ascending order of the reverse priority values, in an ascending order of the number of the precedent commands when the reverse priority values are the same, in an ascending order of line numbers when the reverse priority value and the number of the precedent commands are the same, and in an ascending order of generation times when the reverse priority value, the number of precedent orders and the line number are the same, and in accordance with the order of priority, the first weighting value is determined.

6. The compiler parallelizing schedule method according to claim 5,

wherein in accordance with the order of priority, the first weighting value for the first command is set to a value obtained by subtracting 1 from the number of commands required for issuing the commands within the optimizing target group while taking into consideration the actual issue limitation, and the first weighting value for the commands of the second one and thereafter is set to a value obtained by successively reducing 1 from the value obtained by subtracting 1 from the number of commands.

7. The compiler parallelizing schedule method according to claim 5,

wherein the first weighting value for the precedent commands to the respective commands within the optimizing target group is set to a value that is inherited from the first weighting value for succeeding commands following the precedent commands, and when a plurality of succeeding commands exist, it is set to a value that is inherited from the greatest first weighting value.

8. The compiler parallelizing schedule method according to claim 4,

wherein the second weighting value for the precedent command to the neck command is set to a value that is inherited from the number of commands required for issuing the commands within the optimizing target group corresponding to the neck command while taking into consideration the actual issue limitation.

9. The compiler parallelizing schedule method according to claim 4,

wherein when a new second weighting value is generated resulting from another optimizing target group different from the optimizing target group corresponding to the neck command, the second weighting value for the precedent command to the neck command is set to a value that is obtained by adding the second weighting value.

10. The compiler parallelizing schedule method according to claim 2,

wherein when there is an issue limitation between commands having the same priority value, the number of commands required for issuing the commands having the same priority value in accordance with the actual issue limitation and the number of commands required for issuing the commands having the same priority value on the assumption that there is no issue limitation are found, and the numbers of commands are compared with each other so that, when the number of commands required for issuing the commands having the same priority value in accordance with the actual issue limitation is greater, it is judged that there is a delay due to the issue limitation.

11. The compiler parallelizing schedule method according to claim 3,

wherein in any of cases in which no precedent command exists in commands within an optimizing target group, no command exists between the priority value of the optimizing target group and the priority value of the neck command,

and there is any command that is not a precedent command of an optimizing target group between the priority value of the optimizing target group and the priority value of the neck command, none of the calculating process of the reverse priority values, the weighting process and the calculating process of the new reverse priority values are carried out, and based upon the priority values first found, the order of issue of commands is determined so as to carry out slot mapping of the respective commands.